**BÁO CÁO ĐỒ ÁN 3**



**MÔN HỌC: HỆ ĐIỀU HÀNH**

**Tên: Hà Tấn Linh – 1512284**

**Đỗ Thành Nhơn – 1512387**

**Nguyễn Thành Tân – 1512491**

**Lớp: 15CNTN**

**GVHD: Lê Viết Long**

PHỤ LỤC

[I. Thiết kế các lớp hỗ trợ. 3](#_Toc502317886)

[1. Lớp PCB. 3](#_Toc502317887)

[2. Lớp PTable. 3](#_Toc502317888)

[3. Lớp Sem. 3](#_Toc502317889)

[4. Lớp STable. 4](#_Toc502317890)

[II. Thiết kế các System Call. 4](#_Toc502317891)

[1. System call ExecProcess. 4](#_Toc502317892)

[2. System call ExitProcess. 4](#_Toc502317893)

[3. System call Join Process. 4](#_Toc502317894)

[4. System call CreateSemaphore. 5](#_Toc502317895)

[5. System call Wait. 5](#_Toc502317896)

[6. System call Signal. 5](#_Toc502317897)

[III. Thiết kế chương trình “Sử dụng nước nóng lạnh”. 5](#_Toc502317898)

[1. Chương trình main. 5](#_Toc502317899)

[2. Chương trình sinh viên. 5](#_Toc502317900)

[3. Chương trình void nước. 6](#_Toc502317901)

[IV. Hướng dẫn sử dụng và Demo. 6](#_Toc502317902)

[1. Hướng dẫn sử dụng. 6](#_Toc502317903)

[1.1. Biên dịch chương trình. 6](#_Toc502317904)

[1.2. Gọi chương trình. 6](#_Toc502317905)

[2. Hình ảnh demo. 6](#_Toc502317906)

[2.1. Chương trình in ra hàng chữ AB… xen kẽ. 6](#_Toc502317907)

[2.2. Chương trình Sử dụng máy nóng lạnh. 7](#_Toc502317908)

[V. Đánh giá. 7](#_Toc502317909)

[VI. Tài liệu tham khảo. 7](#_Toc502317910)

# Thiết kế các lớp hỗ trợ.

## Lớp PCB.

* Lớp PCB quản lý một tiến trình, lưu trữ thông tin cần thiết của một tiến trình và các thao tác với tiến trình đó. Các thao tác được cài đặt của PCB là:
* Exec: khởi tạo tiến trình, load tiến trình vào bộ nhớ và bắt đầu thực thi tiến trình đó.
* JoinWait: bước vào trạng thái chờ khi tiến trình đã join vào một tiến trình khác.
* JoinRelease: giải phóng tiến trình và kết thúc trạng thái chờ.
* ExitWait: bước vào trạng thái chờ tiến trình cha cho phép kết thúc.
* ExitRelease: cho phép một tiến trình đã gọi ExitWait kết thúc.
* GetThread: trả về tiến trình mà process đang thực thi.
* SetName và SetExitCode: tương ứng là đặt lại process name và exit code cho tiến trình.
* GetName, GetID, GetParentID, GetNumWait, GetExitCode: trả về các giá trị tương ứng của tiến trình.

## Lớp PTable.

* Lớp Ptable quản lý các tiến trình đang được thực thi trong hệ thống trong một mảng, Ptable còn sử dụng một đối tượng BitMap để đánh dấu các tiến trình trong mảng này. Các thao tác mà Ptable cung cấp là:
* ExecUpdate: Khởi tạo một tiến trình với tên nhận từ tham số name.
* JoinUpdate: Join tiến trình hiện tại vào một tiến trình khác có ID đã biết.
* ExitUpdate: Kết thúc tiến trình hiện tại với exit code tương ứng.
* isExist: kiểm tra một tiến trình có tồn tại hay không.
* Remove: xoá một tiến trình khỏi hệ thống.
* GetName: trả về tên tiến trình khi biết ID của tiến trình.
* GetFreeSlot: trả về một vị trí trống trong mảng quản lý các tiến trình, đồng thời đánh dấu vị trí này đã sử dụng trong BitMap.

## Lớp Sem.

* Lớp Sem quản lý một Semaphore cũng như các thao tác với Semaphore này để đảm bảo quá trình đồng bộ giữa các process. Các thao tác mả lớp Sem cung cấp là:
* GetName: trả về tên của Semaphore đã khởi tạo.
* Wait: Đưa semaphore vào trạng thái chờ.
* Signal: Đưa semaphore trở lại trạng thái bình thường từ trạng thái chờ.

## Lớp STable.

* Lớp Stable quản lý các Semaphore được khởi tạo trong hệ thống bằng một mạng các đối tượng thuộc lớp Sem. Các thao tác mà lớp Stable cung cấp là:
* Create: Tạo một semaphore mới vào mảng các semaphore với tên và giá trị khởi tạo nhận từ tham số name và init.
* Wait: Đưa một Semaphore có tên nhận từ tham số name vào trạng thái chờ.
* Signal: Báo hiệu semaphore có tên nhận từ tham số name kết thúc trạng thái chờ.
* findFreeSlot: Tìm một vị trí trống chưa dùng tới trong mảng các Semaphore.

# Thiết kế các System Call.

## System call ExecProcess.

* Đầu tiên, ExecProcess sẽ lấy địa chỉ của tên process từ thanh ghi r4. Sau đó gọi hàm User2System để copy tên process từ user space sang kernel space. Cuối cùng gọi hàm ExecUpdate của đối tượng ptab với tham số là process name để tạo process và ghi kết quả trả về vào thanh ghi r2.

## System call ExitProcess.

Tương tự như JoinProcess, ExitProcess cũng lấy trị trả về từ thanh ghi r4. Sau đó gọi hàm ExitUpdate để kết thúc tiến trình và cuối cùng ghi kết quả trả về vào thanh ghi r2.

## System call Join Process.

* JoinProcess đầu tiên sẽ lấy Process ID của process cần join từ thanh ghi r4. Sau đó gọi hàm JoinUpdate của đối tượng ptab để join vào process tương ứng. Cuối cùng ghi kết quả trả về vào thanh ghi r2.

## System call CreateSemaphore.

* Đầu tiên, CreateSemaphore sẽ lấy địa chỉ tên semaphore ở user space từ thanh ghi r4 và giá trị khởi tạo của semaphore từ thanh ghi r5. Sau đó gọi hàm User2System để copy tên từ user space sang kernel space. Cuối cùng CreateSemaphore sẽ gọi hàm Create của đối tượng stab để tiến hành tạo semaphore với các tham số tương ứng và ghi giá trị trả về vào thanh ghi r2 trước khi kết thúc system call.

## System call Wait.

* Wait trước tiên sẽ đọc địa chỉ tên semaphore của user từ thanh ghi r4. Sau đó gọi User2System để copy tên vào kernel space từ user space. Cuối cùng Wait gọi hàm Wait của đối tượng stab với tham số là tên semaphore để thực hiện yêu cầu từ user và ghi trị trả về tương ứng vào thanh ghi r2.

## System call Signal.

* Các công việc của Signal cũng hoản toàn tương tự như system call Wait đã trình bày ở trên, chỉ khác một điểm duy nhất là thay vì gọi hàm Wait thì syscall Signal gọi hàm Signal của đối tượng stab thuộc kiểu Stable.

# Thiết kế chương trình “Sử dụng nước nóng lạnh”.

Đối với bài toán vòi nước, chúng ta sử dụng 3 chương trình riêng biệt phối hợp với nhau để giải.

## Chương trình main.

* Chương trình main: là chương trình khởi đầu, nó sẽ đọc file input.txt, với mỗi bộ dữ liệu, main sẽ copy dòng dữ liệu đó vào file turn.txt. Sau đó gọi syscall ExecProcess để tạo một tiến trình sinhvien. Tiếp theo gọi syscall JoinProcess để join vào tiến trình vừa được tạo và chờ nó kết thúc. Cuối cùng main nối kết quả nhận được từ sinhvien trong file rslt.txt vào ouput.txt.

## Chương trình sinh viên.

* Chương trình sinhvien: là chương trình quản lý các lượt sinh viên đến sử dụng vòi nước. sinhvien đầu tiên sẽ gọi syscall ExecProcess để tạo một process voinuoc. Sau đó với mỗi số nguyên trong file turn.txt (tương ứng với một lượt sinh viên) sinhvien sẽ ghi số nguyên đó vào file require.txt sau đó gọi syscall Signal để báo hiệu cho voinuoc làm việc và gọi syscall Wait để chờ voinuoc kết thúc làm việc và xử lý kết quả nhận được từ file tap.txt.

## Chương trình void nước.

* Chương trình voinuoc: là chương trình điều phối các lượt sinh viên sử dụng 2 vòi nước. voinuoc sẽ lặp lại liên tục quá trình gọi syscall Wait để chờ thông báo làm việc từ sinhvien, đọc số lượng nước yêu cầu trong file tap.txt. Đọc trạng thái của 2 vòi nước từ file first.txt và second.txt. Sau đó voinuoc sẽ chọn vòi nước đang sẵn sàng tại thời điểm đó, cập nhật vòi nước và ghi tên vòi nước vào file tap.txt và gọi syscall Signal để báo cho sinhvien là nó đã kết thúc phiên làm việc.

# Hướng dẫn sử dụng và Demo.

## Hướng dẫn sử dụng.

### Biên dịch chương trình.

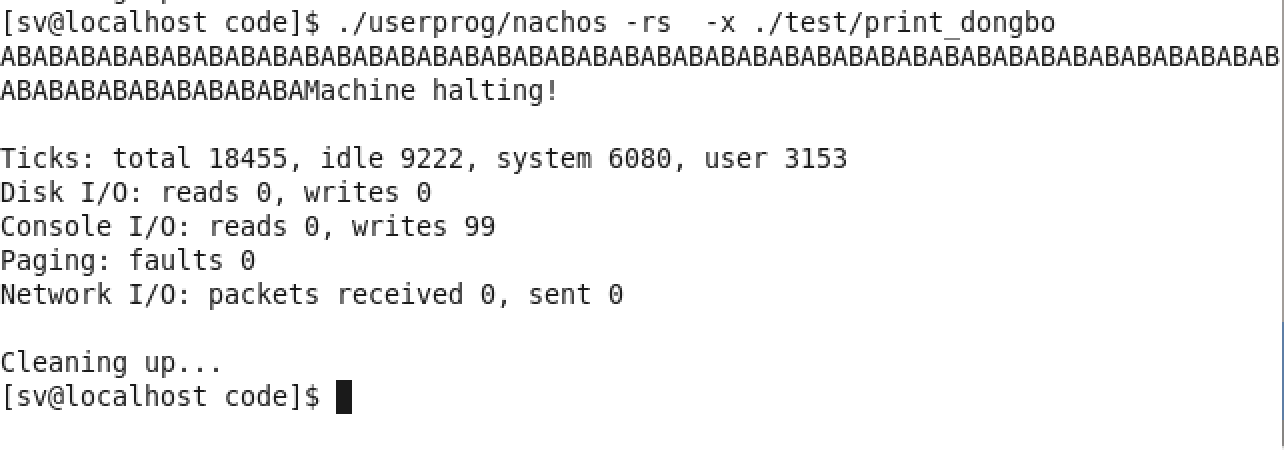
* Mở Teminal ở thư mục code.
* Gõ: make.

### Gọi chương trình.

* Mở Teminal ở thư mục code.
* Gõ lệnh sau: ./userprog/nachos –rs –x ./test/<ten\_chuong\_trinh>

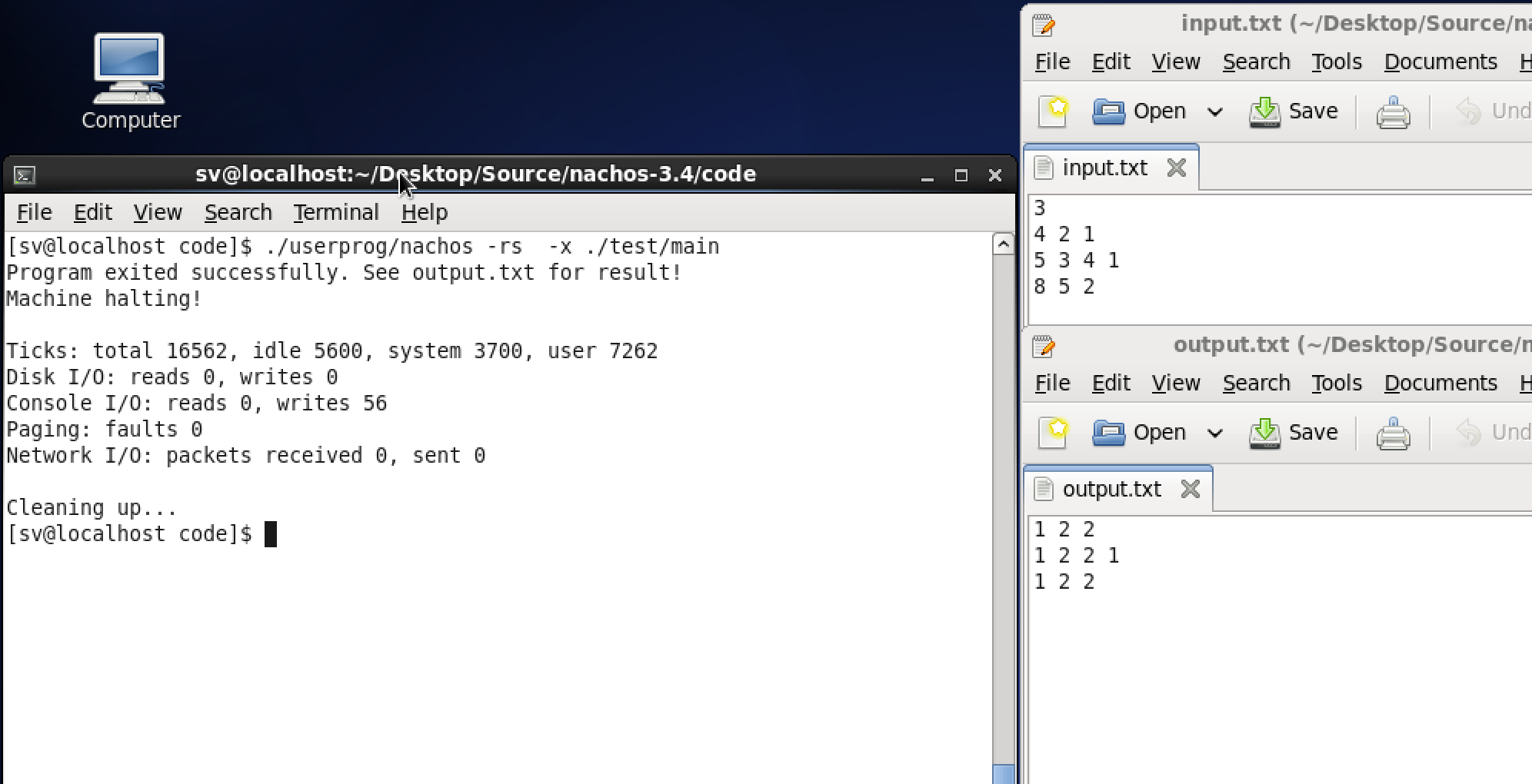
## Hình ảnh demo.

### Chương trình in ra hàng chữ AB… xen kẽ.



Hình 1: Chương trình in A và B xen kẽ

### Chương trình Sử dụng máy nóng lạnh.



Hình 2: Chương trình sử dụng máy nóng lạnh

# Đánh giá.

* Làm được: Cài đặt được các lớp hỗ trợ và các system calls như trong yêu cầu đồ án. Cài đặt được các chương trình giải quyết bài toán vòi nước và đảm bảo các chương trình hoạt động một cách đồng bộ.
* Chưa làm được: Khi dữ liệu trong input.txt có quá nhiều trường hợp thì không đủ bộ nhớ để chứa tất cả các process nên sẽ bị assertion failed.
* Mức độ hoàn thành: 100% yêu cầu đồ án.

# Tài liệu tham khảo.

*Tài liệu hướng dẫn mà giảng viên cung cấp trên Moodle.*